Received January 31, 2021, accepted February 23, 2021, date of publication February 26, 2021, date of current version March 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3062405

eWB: Event-Based Weight Binarization Algorithm for Spiking Neural Networks

DOHUN KIM[1,2], GUHYUN KIM[3], CHEOL SEONG HWANG[1,2], AND DOO SEOK JEONG[3], (Member, IEEE)

[1]Department of Material Science and Engineering, Seoul National University, Seoul 08826, Republic of Korea
[2]Inter-University Semiconductor Research Center, Seoul National University, Seoul 08826, Republic of Korea
[3]Division of Materials Science and Engineering, Hanyang University, Seoul 04763, Republic of Korea

Corresponding author: Doo Seok Jeong (dooseokj@hanyang.ac.kr)

This work was supported by the research grant of the National Research Foundation of Korea under Grant NRF-2019R1C1C1009810 and by the Future Semiconductor Device Technology Development Program (20012002) funded by Ministry of Trade, Industry & Energy and Korea Semiconductor Research Consortium.

ABSTRACT Learning binary weights to minimize the difference between target and actual outputs can be considered as a parameter optimization task within the given constraints, and thus, it belongs to the application domain of the Lagrange multiplier method (LMM). Based on the LMM, we propose a novel event-based weight binarization (eWB) algorithm for spiking neural networks (SNNs) with binary synaptic weights (−1, 1). The algorithm features (i) event-based asymptotic weight binarization using local data only, (ii) full compatibility with event-based end-to-end learning algorithms (e.g., event-driven random backpropagation (eRBP) algorithm), and (iii) the capability to address various constraints (including the binary weight constraint). As a proof of concept, we combine eWB with eRBP (eWB-eRBP) to obtain a single algorithm for learning binary weights to generate correct classifications. Fully connected SNNs were trained using eWB-eRBP and achieved an accuracy of 95.35% on MNIST. To the best of our knowledge, this is the first report on completely binary SNNs trained using an event-based learning algorithm. Given that eRBP with full-precision (32-bit) weights exhibited 97.20% accuracy, the binarization comes at the cost of an accuracy reduction of approximately 1.85%. The python code is available online: https://github.com/galactico7/eWB.

INDEX TERMS Event-based weight binarization, event-driven learning algorithm, Lagrange multiplier method, spiking neural networks.

I. INTRODUCTION

There has been growing interest in fast, efficient, and compact neuromorphic computing for high-performance processing of large amounts of data for on-chip learning. Spiking neural networks (SNNs) are a promising model for energy-efficient neuromorphic computing [1]–[3]. Their energy efficiency is mainly due to the sparse event-based asynchronous data processing and learning weights, as opposed to the case for deep neural networks (DNNs), which utilize error-backpropagation algorithms (BP) for layer-wise synchronous weight updates in dedicated learning phases [1]. Further efficiency improvements are gained when SNNs are implemented on dedicated neuromorphic hardware [4], [5]. To date, several event-based learning algorithms have been proposed, including spike timing-dependent plasticity (STDP) [6]–[8], event-driven random backpropagation (eRBP) [9], sequence-predicting SNN [10], ReSuMe [11], tempotron [12], and Spikeprop [13]. However, because most of these event-based algorithms use multi-bit weights, their hardware implementation requires large on-chip memory capacity and intensive computing power, which degrades their energy efficiency. Weight quantization has been considered to address this issue, for example, in recent STDP-based algorithms with quantized weights [7], [8], [14]–[16]. However, achieving a competitive classification accuracy commonly requires (i) a large number of trainable parameters, especially those related to hidden neurons, (ii) an inhomogeneous learning framework to consider BP and STDP separately, and (iii) multi-bit weights for output evaluation.

Learning binary weights is an extreme case of weight quantization. The use of 1-bit weights significantly reduces

The associate editor coordinating the review of this manuscript and approving it for publication was Pasquale De Meo.

on-chip memory usage considering the $O(n^2)$ memory complexity of synapses. Additionally, the leaky integrate-and-fire (LIF) model with an exponentially decaying synaptic current kernel involves the multiplication of weights and low-pass filtered spikes [17]. Thus, learning binary weights avoids multiplication and significantly reduces power consumption and processing time.

To this end, we propose an event-driven weight binarization (eWB) algorithm to learn binary weights $(-1, 1)$ in an event-based manner. The eWB algorithm uses the Lagrange multiplier method (LMM) based on a Lagrange function that combines a conventional loss function and constraints on binary weights. Each synapse is given a binary weight constraint function and a Lagrange multiplier. The binarization of each weight is independent of the variables in the other synapses. This ensures the locality of eWB. Because a conventional loss function is also used, eWB is not a standalone learning algorithm. Instead, it requires an additional event-based supervised learning algorithm based on a loss function such as eRBP. As a proof of concept, we combine eWB and eRBP (eWB-eRBP) to train fully connected multilayer SNNs on MNIST. The results demonstrate successful weight binarization at the cost of an accuracy reduction by approximately 1.85%.

The primary contributions of this study are as follows:

- We introduce a novel weight binarization algorithm (eWB), which is an event-driven algorithm with locality.
- We elaborate on the scaling of eWB to eRBP (eWB-eRBP).
- We evaluate the performance of eWB-eRBP in training fully connected multilayer SNNs on MNIST.
- The code for eWB-eRBP is available on-line (https://github.com/galactico7/eWB).

Section II (Related work) briefly overviews several relevant studies. Section III introduces the basic principles of LMM (Section III. A), eWB (Section III. B), and eWB-eRBP (Section III. C). Section IV presents the experimental results (prediction accuracy, weight binarization and efficiency analysis) on MNIST. A comparison with state-of-the-art weight binarization in SNNs is given in Section V. Finally, Section VI concludes this study.

II. RELATED WORK

To date, the proposed methods for learning binary weights in SNNs can be mainly classified into two groups: (i) conversion of binary neural networks (BNNs) [18] or binary weight networks (BWNs) [19] into SNNs and (ii) probabilistic learning. The first approach is motivated by recent successes in weight binarization in DNNs [18]–[21]. Lu and Sengupta [22] proposed a method to map BWNs onto VGG-15-like SNNs with perfect integrate-and-fire (IF) neurons. The IF neuron is considered to encode an input spike train as a firing rate, thereby ensuring its similarity to a rectified linear unit (ReLU).

The second approach uses weights that probabilistically toggle between the binary weights 0 and 1. However, it is common to use auxiliary variables to determine the probability of weight updates. Suri *et al.* [14] proposed an STDP-based stochastic algorithm using binary weights. Potentiation $(0\rightarrow 1)$ and depression $(1\rightarrow 0)$ occur with probabilities based on the temporal order of presynaptic and postsynaptic spikes. Nevertheless, a challenge in STDP is its inability to scale to deep SNNs. STDP is hence commonly limited to shallow SNNs. Yousefzadeh *et al.* [7] also proposed an STDP-based stochastic algorithm to learn binary weights. The algorithm requires additional operations such as weight normalization and threshold adjustment for individual neurons. The considered network is shallow (one feature extraction layer and one classifier) because of the aforementioned limitation of STDP. Additionally, it does not support end-to-end training. Srinivasan and Roy [8] proposed the Hybrid-STDP (HB-STDP) algorithm based on probabilistic STDP. Notably, HB-STDP includes a dead zone in the STDP time window, in which neither potentiation nor depression is allowed. HB-STDP captures temporally correlated inputs by preventing excessive potentiation and depression; however, its low accuracy is a challenge.

The learning of low-precision weights in SNNs as generative models is a subject of interest. Stromatias *et al.* [23] tailored contrastive divergence (CD) to deep belief networks with spiking neurons. The original double-precision floating-point weights are converted to a lower precision floating-point format to reduce memory consumption. However, the reduction in precision comes with the cost of significant performance degradation. Neftci *et al.* [16] proposed the event-driven CD (eCD) algorithm to train restricted Boltzmann machines with spiking neurons. In eCD, the weight update is fine-tuned by STDP. Low-precision (down to 2-bit) weights were tested; however, a significant reduction in accuracy was unavoidable.

The eWB algorithm adopts an approach that parameterizes the degree of binary-constraint fulfillment and asymptotically optimizes the degree upon the occurrence of events, whereas the aforementioned precision-reduction methods merely round the full-precision weights. Therefore, eWB cannot be classified as any of the aforementioned approaches.

III. eWB ALGORITHM

A. LAGRANGE MULTIPLIER METHOD

LMM is a strategy to solve general nonlinear programming problems (NLPs) [24]. An NLP is an optimization problem whose optimal solution is determined by nonlinear constraints in conjunction with a nonlinear objective function. In the minimization problem, a general continuous equality-constrained NLP can be stated as

$$\begin{aligned} &\text{minimize } l(\boldsymbol{w}); \boldsymbol{w} = [w_1, w_2, \cdots, w_n] \\ &\text{subject to } \boldsymbol{g}(\boldsymbol{w}) = 0; \quad \boldsymbol{g} = [g_1, g_2, \cdots, g_m]. \end{aligned}$$

The LMM calculates the local maxima or minima of the objective function within the given equality constraints. The Lagrange function L for the objective function l and

constraints $\boldsymbol{g}$ is expressed as

$$L(\boldsymbol{w}, \boldsymbol{\lambda}) = l(\boldsymbol{w}) + \lambda \cdot \boldsymbol{g}(\boldsymbol{w}), \tag{1}$$

where $\boldsymbol{\lambda}$ $(= [\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \cdots, \boldsymbol{\lambda_m}])$ is a vector of Lagrange multipliers with one multiplier for each of the m constraints. If $\boldsymbol{w}^*$ is a local extremum point of the objective function $l(\boldsymbol{w})$ subject to $\boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0}$, the following equalities hold:

$$\begin{cases} \nabla_{\boldsymbol{w}} l(\boldsymbol{w}) + \boldsymbol{\lambda} \cdot \nabla_{\boldsymbol{w}} \boldsymbol{g}(\mathbf{w}) = \boldsymbol{0} \text{ at } \boldsymbol{w}^* \\ \boldsymbol{g}(\boldsymbol{w}^*) = \boldsymbol{0}. \end{cases} \tag{2}$$

This optimal solution $\boldsymbol{w}^*$ can be calculated from the gradient of Eq. (1):

$$\begin{aligned} &\nabla_{\boldsymbol{w},\boldsymbol{\lambda}} \boldsymbol{L}(\boldsymbol{w}, \boldsymbol{\lambda}) \\ &\quad = \nabla_{\boldsymbol{w}} [l(\boldsymbol{w}) + \lambda \cdot \boldsymbol{g}(\boldsymbol{w})] + \nabla_{\boldsymbol{\lambda}} [l(\boldsymbol{w}) + \lambda \cdot \boldsymbol{g}(\boldsymbol{w})]. \end{aligned}$$

The condition $\nabla_{\boldsymbol{w},\boldsymbol{\lambda}} L(\boldsymbol{w}, \boldsymbol{\lambda}) = 0$ is equivalent to the following conditions:

$$\begin{cases} \nabla_{\boldsymbol{w}} [l(\boldsymbol{w}) + \boldsymbol{\lambda} \cdot \boldsymbol{g}(\boldsymbol{w})] = \nabla_{\boldsymbol{w}} l(\boldsymbol{w}) + \boldsymbol{\lambda} \cdot \nabla_{\boldsymbol{w}} \boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0} \\ \nabla_{\boldsymbol{\lambda}} [l(\boldsymbol{w}) + \boldsymbol{\lambda} \cdot \boldsymbol{g}(\boldsymbol{w})] = \boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0}, \end{cases}$$

which are identical to the conditions in Eq. (2). Therefore, the optimal point $\boldsymbol{w}^*$ leads to

$$\nabla_{\boldsymbol{w},\boldsymbol{\lambda}} L(\boldsymbol{w}, \boldsymbol{\lambda}) = 0. \tag{3}$$

The solution to Eq. (3) can be calculated using a basic differential multiplier method [25], in which the optimal $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ are calculated using a gradient descent and ascent method, respectively, i.e.,

$$\begin{cases} \boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \eta_w \nabla L_{\boldsymbol{w}}(\boldsymbol{w}^k, \boldsymbol{\lambda}^k) \\ \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \eta_\lambda \boldsymbol{g}\left(\boldsymbol{w}^k\right), \end{cases} \tag{4}$$

where η_w and η_λ are learning rates. Note that this method is compatible with the BP in DNNs if the model parameters $\boldsymbol{w}$ are updated using a gradient descent method but with the Lagrange function L rather than the loss function l taken as the objective function.

B. eWB ALGORITHM

The eWB algorithm is based on LMM with binary weight constraints. The key feature of this algorithm is that the configuration of binary weights over the network is subject to optimization, unlike common weight binarization methods that force the weights to assume binary values using particular binarization functions. Because each synapse is given a binary weight constraint and a Lagrange multiplier, the total numbers of constraints and Lagrange multipliers are equal to the number of synapses in the SNNs. Here, we select the binary weight constraint g for a given synapse as

$$g_i(w_i, t) = \left(1 - w_i^2\right) s_i(t), \tag{5}$$

which is zero if $w_i = \pm 1$. We introduce the spike function s_i, which yields one when a presynaptic (or postsynaptic) spike occurs and zero otherwise, and hence enables event-based asymptotic binarization. The loss function $l(\boldsymbol{w},t)$ should be chosen to enable event-based weight updates and satisfy

$$\frac{\partial l}{\partial w_i} = h(\boldsymbol{w}, t)\, s_i(t),$$

where h is the product of the backpropagating error and the derivative of postsynaptic output. The Lagrange function L is given by

$$L(\boldsymbol{w}, \boldsymbol{\lambda}, t) = l(\boldsymbol{w}, t) + \sum\nolimits_i \lambda_i \left(1 - w_i^2\right) s_i(t).$$

Consequently, the weight and Lagrange multiplier are updated upon a presynaptic (or postsynaptic) spike of synapse w_i conforming to Eq. (4):

$$\begin{cases} w_i \leftarrow w_i - \eta_w \dfrac{\partial L}{\partial w_i} \\ \lambda_i \leftarrow \lambda_i + \eta_\lambda \dfrac{\partial L}{\partial \lambda_i}. \end{cases} \tag{6}$$

These real-valued weights and Lagrange multipliers are stored in the memory for successive event-based updates. However, during training with eWB, both signal forward propagation and error backpropagation use the forced-to-be-binary weights w^{b}:

$$w_i^{\mathrm{b}} = \mathrm{Sign}(w_i) = \begin{cases} +1 & \text{if } w_i \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{7}$$

This avoids the multiplication of real-valued weights and low-pass filtered spikes in the LIF model, thereby significantly reducing the computational complexity and, thus, the power consumption.

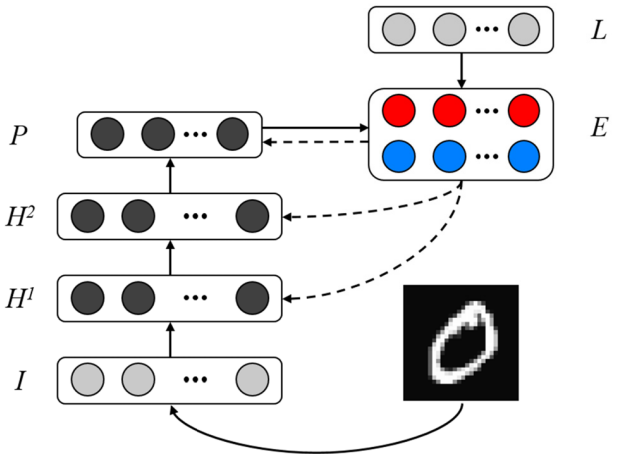

FIGURE 1. SNN architecture for eRBP. The error-coding layer (E) consists of two error-coding neurons for each label dimension that encode false positive and negative errors between labels (L) and predictions (P). During training, each of the hidden (in H^1 and H^2) and prediction (in P) neurons receives random feedback from the error neurons with fixed random weights (dashed arrows). The input layer is indicated by I.

C. eWB-eRBP ALGORITHM

As a proof of concept, we chose eRBP for combination with eWB (eWB-eRBP). As shown in Fig. 1, eRBP is a three-factor rule based on (i) presynaptic events, (ii) approximated gradients of postsynaptic activation, and (iii) error

signals through random feedback channels [9]. The eRBP algorithm is elaborated in the Appendix. Accordingly, the loss function l is considered to be the mean-squared difference between the target and actual outputs. The eRBP algorithm defines the derivative of the loss function with respect to the weight w_{ij} between the presynaptic neuron j and postsynaptic neuron i as

$$\frac{\partial l}{\partial w_{ij}} \stackrel{\text{def}}{=} T_i(t)\Theta(I_i)s_j^{pre}(t), \tag{8}$$

where T_i is a random backpropagation error for the weight update. The gradient of the postsynaptic activation is approximated by a boxcar function Θ with two transition points ($b_{\min}$ and $b_{\max}$), which is a function of the synaptic current I_i:

$$\Theta\left(I_i\right) = \begin{cases} 1 & \text{if } b_{min} < I_i < b_{max} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The spike train of the presynaptic neuron j is denoted by s_j^{pre}. This term allows weight update upon the occurrence of presynaptic events only; therefore, eRBP is a presynaptic event-based learning algorithm. We tailor the binary weight constraint function g to eRBP such that

$$g\left(w_{ij}\right) = (1-w_{ij}^2)\Theta(I_i)s_j^{pre}(t). \tag{10}$$

The modification can be seen by comparing with Eq. (5). For compatibility with eRBP, eWB is also assumed to be driven by presynaptic events. Additionally, we incorporate the approximated postsynaptic activation gradient Θ into the constraint function. Θ is included in the constraint function to synchronize the weight update for reducing the constraint function g with the update for reducing the loss function l. Otherwise, it may be possible that only the constraint g is reduced during the weight update, irrespective of the loss function l, especially when $\Theta = 0$.

The derivative of the boxcar function Θ in Eq. (9) is zero except for the two transition points ($b_{\min}$ and $b_{\max}$), which are singular points. Thus, the boxcar function is non-differentiable. As a workaround, we assume that the synaptic current I_i avoids these transition points when presynaptic events occur, which is highly probable because the probability of the current being equal to either of the two particular values is extremely low. Therefore, the following equation holds:

$$\begin{aligned} \frac{\partial g}{\partial w_{ij}} &= -2w_{ij}\Theta\left(I_i\right)s_j^{pre}\left(t\right) + \left(1-w_{ij}^2\right)s_j^{pre}\left(t\right)\frac{\partial\Theta\left(I_i\right)}{\partial I_i}\frac{\partial I_i}{\partial w_{ij}} \\ &\approx -2\mathrm{w_{ij}}\Theta\left(I_i\right)s_j^{pre}\left(t\right). \end{aligned} \tag{11}$$

Using Eqs. (6), (8), and (11), we evaluate the updates on the weight w_{ij} and the Lagrange multiplier λ_{ij}:

$$\begin{cases} \Delta w_{ij} = -\eta_w\dfrac{\partial L}{\partial w_{ij}} = -\eta_w\left(\dfrac{\partial l}{\partial w_{ij}} + \lambda_{ij}\dfrac{\partial g}{\partial w_{ij}}\right) \\ = -\eta_w\left(T_i\left(t\right) - 2\lambda_{ij}w_{ij}\right)\Theta\left(I_i\right)s_j^{pre}\left(t\right) \\ \Delta\lambda_{ij} = \eta_\lambda\dfrac{\partial L}{\partial\lambda_{ij}} = \eta_\lambda g = \eta_\lambda\left(1-w_{ij}^2\right)\Theta\left(I_i\right)s_j^{pre}\left(t\right). \end{cases} \tag{12}$$

As highlighted in the previous section, the weights for the forward paths (to calculate current input I_i) and backward paths (to calculate T_i) are forcibly binarized (Eq. (7)) to reduce the hardware computing workload.

The weights are initialized using the Xavier uniform initialization [26], whereas the Lagrange multipliers are initialized to zero. We confine each weight to between -1 and 1 by projecting w to -1 (1) when the updated weight is smaller than -1 (larger than 1). This weight clipping prevents unlimited weight growth. The eWB-eRBP algorithm is given in pseudocode in Algorithm 1.

Algorithm 1 eWB-eRBP Algorithm.

Initialize $\boldsymbol{w}$, $\boldsymbol{\lambda}$
while *True* **do**
  $\boldsymbol{w}^b \leftarrow$ Binarize($\boldsymbol{w}$)
  for k $\in$ {presynaptic event indices $\boldsymbol{s}^{pre}$} **do**
    if $b_{\min} < I < b_{\max}$ **then**
      $\boldsymbol{w} \leftarrow Clip\{\boldsymbol{w} - \eta_w\nabla_{\boldsymbol{w}}L, -1, 1\}$
      $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \eta_\lambda\nabla_{\boldsymbol{\lambda}}L$
    end if
  end for
return $\boldsymbol{w}^b$

D. NON-OPTIMAL WEIGHT BINARIZATION METHOD

The defining feature of eWB is the optimization of the binary weight distribution over the SNN. To highlight the performance of eWB, we compare eWB-eRBP with eRBP in conjunction with forced-to-be-binary weights conforming to Eq. (7), referred to as fWB-eRBP. Note that fWB stands for forced weight binarization. In fWB-eRBP, the binary weight distribution is non-optimal, and the real-valued weights are optimized using the loss function l only. In eWB-eRBP, the real-valued weights are used for weight update only, and the signal forward propagation and error backpropagation use the binarized weights given by Eq. (7) instead. The fWB-eRBP algorithm is given in pseudocode in Algorithm 2.

Algorithm 2 fWB-eRBP Algorithm.

Initialize $\boldsymbol{w}$
while *True* **do**
  $\boldsymbol{w}^b \leftarrow$ Binarize($\boldsymbol{w}$)
  for k $\in$ {presynaptic spike indices $\boldsymbol{s}^{pre}$} **do**
    if $b_{\min} < I < b_{\max}$ **then**
      $\boldsymbol{w} \leftarrow Clip\{\boldsymbol{w} - \eta_1\nabla_{\boldsymbol{w}}f, -1, 1\}$
    end if
  end for
return $\boldsymbol{w}^b$

IV. RESULTS

We trained three types of fully connected SNNs (784-h-h-10; h = 200, 500, and 1000) on MNIST. One training epoch consisted of 60,000 full training data that were selected

randomly and input into the SNN. The intensity of each pixel in each hand-written digit image was encoded as the firing rate of input spikes (10–265 Hz) in proportion to the intensity. Note that even blank pixels were encoded at 10 Hz to serve as low-frequency background noise. Each image was shown to the SNN for 200 ms. To avoid interference from the previous training image, all neuronal variables were reset to zero before the onset of the current training image. The classification accuracy was evaluated once every 500 training data using the 10,000 test data. The predicted output was identified by counting the number of spikes from each output neuron for 200 ms. The parameters used are listed in Table 1.

TABLE 1. Parameters for simulations.

Symbol	Explanation	Value
N_d	Number of data neurons	784
N_h	Number of hidden neurons	200,500,1000
N_l	Number of label neurons	10
N_{E+}	Number of positive error neurons	10
N_{E-}	Number of negative error neurons	10
N_p	Number of prediction neurons	10
τ_{refr}	Refractory period	4 ms
τ_{syn}	Synaptic time constant	4 ms
g_V	Leak constant state V	1 nS
g_U	Leak constant state U	5 nS
C	Membrane capacitance	1 pF
V_{th}	Threshold for spikes	1.1 V
w^E	Fixed weight	1 nA
b_{min}, b_{max}	Boxcar function constants	-25, 25 nA
η_1	Learning rate	$2e^{-4}$
η_2	Lagrange multiplier step-size parameter	$2e^{-7}$

TABLE 2. Classification accuracy on the MNIST dataset for eRBP, eWB-eRBP, and fWB-eRBP after 25 epochs.

	Classification accuracy (%)		
Network	eRBP	eWB-eRBP	fWB-eRBP
784-200-200-10	96.32	93.81	93.59
784-500-500-10	96.92	95.05	94.50
784-1000-1000-10	97.20	95.35	94.77

A. CLASSIFICATION ACCURACY

We used the three aforementioned algorithms (eRBP, eWB-eRBP, and fWB-eRBP) to train the three SNNs (784-*h*-*h*-10; $h = 200$, 500, and 1000). The final classification accuracy for each case was measured after the 25th training epoch (Table 2). For all three algorithms, the 784-1000-1000-10 SNN achieved the best accuracy. The accuracy evolution of this SNN for each algorithm is shown in Fig. 2. It is noted that weight binarization using either algorithm results in the loss of classification accuracy.

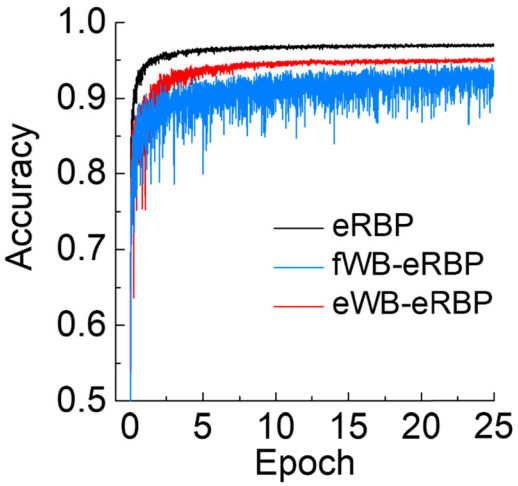

FIGURE 2. Classification accuracies of 784-1000-1000-10 SNNs on MNIST that were trained using eRBP, fWB-eRBP, and eWB-eRBP.

Nevertheless, eWB-eRBP outperforms fWB-eRBP in terms of the loss for all SNNs. For example, for 784-1000-1000-10, the losses for eWB-eRBP and fWB-eRBP are 1.85% and 2.43%, respectively. This highlights the importance of optimal weight binarization for inference. Fig. 2 also shows that the fluctuations in accuracy over the inference period for eWB-eRBP are negligible compared with those for fWB-eRBP. This stability results from eWB asymptotically driving the real-valued weights toward the binary weights during training. Thus, the forced-to-be-binary weights conforming to Eq. (7) that are used for inference negligibly alter the accuracy over successive inference periods, particularly when the weights are close to binary values.

B. WEIGHT BINARIZATION

To evaluate the degree of weight binarization during training, we introduce a constraint failure score (CFS) for a real-valued weight matrix $\boldsymbol{w} \in \mathbb{R}^{N\times M}$ as follows:

$$\mathrm{CFS} = 1 - \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij}^2.$$

Therefore, when all weights are binarized, the CFS equals zero. We monitored the change in CFS over the training epoch (using eWB-eRBP or fWB-eRBP) for the 784-1000-1000-10 SNN. There are three weight matrices: $\boldsymbol{w}w^{(hi)}$ (between the first hidden layer and the input layer), $\boldsymbol{w}w^{(hh)}$ (between the second and first hidden layers), and $\boldsymbol{w}w^{(oh)}$ (between the output layer and the second hidden layer). The changes in CFS for these matrices are shown in Figs. 3(a), (b), and (c), respectively. The CFS for eWB-eRBP asymptotically decreases to zero, ensuring successful weight binarization. For eWB-eRBP, the distributions of the trained real-valued weights in the weight matrices $\boldsymbol{w}w^{(hi)}$, $\boldsymbol{w}w^{(hh)}$, and $\boldsymbol{w}w^{(oh)}$ are plotted in Figs. 3(d), (e), and (f), respectively. These distributions are compared with the distributions of the initial weights. Considering a weight w ($|w| > 0.9$) to be fully binarized, the proportions of such fully binarized weights

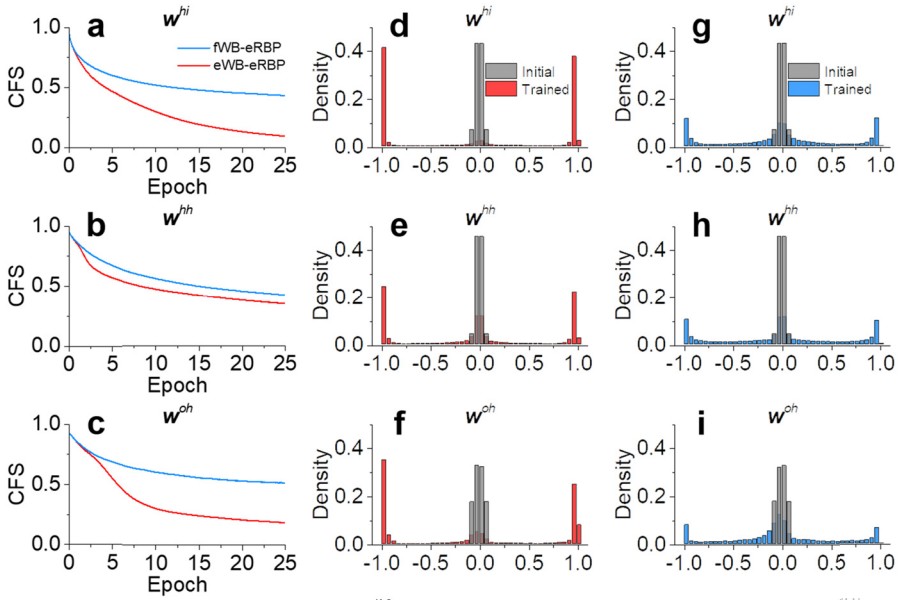

FIGURE 3. Weight distribution of eWB-eRBP and fWB-eRBP for $w^{(hi)}$ (between the first hidden layer and input layer), $w^{(hh)}$ (between the second and first hidden layers), and $w^{(oh)}$ (between the output layer and the second hidden layer). (a)–(c) Changes in CFS over epoch for eWB-eRBP and fWB-eRBP. The weight distribution of the initial and trained real-valued weights for (d)–(f) eWB-eRBP and (g)–(i) fWB-eRBP.

are 84.7%, 53.9%, and 72.9% in $ww^{(hi)}$, $ww^{(hh)}$, and $ww^{(oh)}$, respectively. The main cause of imperfect binarization is discussed in the following section. In contrast, the weight distribution for fWB-eRBP is rather diffusive over the entire weight range [Figs. 3(g)–(i)]. Consequently, the proportions of fully binarized weights after training are 30.0%, 27.1%, and 18.0% for $ww^{(hi)}$, $ww^{(hh)}$, and $ww^{(oh)}$, respectively.

C. COMPUTATIONAL COMPLEXITY

Although the eWB algorithm is proposed for neuromorphic processors, for the moment, neuromorphic processors that serve as platforms for algorithm studies with high degrees of freedom are not available at hand. Instead, we used a GPU workstation (CPU: Intel Xeon Silver 4110 2.10GHz, GPU: RTX 2080 Ti). The algorithm was implemented in Python (the code is available online: https://github.com/galactico7/eWB). Because eWB is not a standalone learning algorithm, we measured the time complexity of eWB from the difference in time complexity between RBP and eWB-eRBP. The eRBP and eWB-eRBP algorithms applied to a 784-500-500-10 SNN on MNIST for 25 learning epochs, yielding a wall-clock time of 4.99E5 s and 6.27E5 s, respectively (Table 3). The additional wall-clock to eRBP (1.28E5 s) arose from eWB. Additionally, we measured the space complexity for eRBP and eWB-eRBP, 366.0 and 368.0 MB, respectively. The MNIST dataset occupies 360.0 MB, so that eRBP and eWB-eRBP occupy 6.0 and 8.0 MB, respectively.

This computational complexity should differ from that measured on neuromorphic hardware, particularly, wall-clock time. However, for digital neuromorphic hardware,

TABLE 3. Time and space complexities for eRBP and eWB-eRBP.

	eRBP	eWB-eRBP
Structure	784-500-500-10	
Dataset	MNIST	
Accuracy	96.92 (25 epochs)	95.05 (25 epochs)
Wall-clock time (s)	4.99E5	6.26E5
Memory usage (MB)	366.0	368.0

the measure of memory usage (for the dataset, parameters, and hyper-parameters) may hold because the same data should be stored in on-chip memory. Additionally, given that most studies on SNN learning algorithms are conducted on general-purpose workstations for the moment, the complexity measure on general-purpose workstations is practically helpful.

Akin to multiply-accumulate operations (MACs) for deep learning implemented in general-purpose hardware, synaptic operations (SynOps) in neuromorphic hardware are known to consume considerable power, so that the number of SynOps can be a relative measure of energy-efficiency for learning. We evaluated the number of SynOps required for training a 784-500-500-10 SNN on MNIST using eWB-eRBP. The SNN was trained for 25 epochs in aggregate. Fig. 4 shows the evaluated number of SynOps and classification accuracy for each learning epoch.

For a comparison with binarized neural network (BNN) [18], we measured the number of MACs required for training a 784-500-500-10 BNN on MNIST. Each MNIST image was pre-binarized to ± 1 using the sign function.

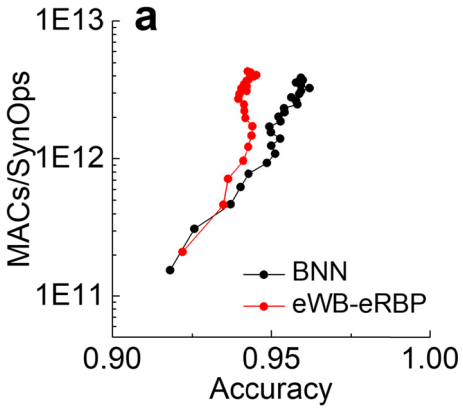

FIGURE 4. Efficiency in learning. (a) Number of SynOps for a 784-500-500-10 SNN with eWB-eRBP algorithm. BNN (784-500-500-10 network) is compared with the eWB-eRBP in terms of the number MACs required for reaching a given accuracy for the MNIST learning task.

We used Batch Normalization with a minibatch size of 100. The square hinge loss was minimized using Adam optimizer. We employed an exponentially decaying global learning rate and Glorot initialization. Dropout layers were deployed to regularize the BNN. The evaluation results are co-plotted in Fig. 4, indicating that both networks require similar same numbers of operations to reach an accuracy approximately 0.93.

The comparison indicates similar operational complexity for both cases. Yet, power-efficiency for eWB-eRBP likely outperforms BNN when implemented in neuromorphic hardware. Power-efficiency is the defining feature of neuromorphic hardware. For instance, Loihi (digital neuromorphic processor) [27] highlights high power-efficiency, approximately 300 times that of graphics processing units [28]. Thus, we expect a two orders of magnitude increase in power-efficiency when eWB-eRBP is embedded in neuromorphic hardware.

V. DISCUSSION

Generally, event-based learning algorithms update a weight only if a presynaptic or postsynaptic event (local to the synapse) occurs, unlike BP, which updates all weights layer-wise. Specifically, eWB-eRBP addresses only the synapses that satisfy the two conditions of (i) presence of presynaptic spike, and (ii) non-zero boxcar function of the postsynaptic activation, as described in Eq. (12). During training, several synapses were inactive (and their presynaptic neurons quiescent), and thus, they maintained their initial weights until the end of training. The high proportion of fully binarized weights in $\boldsymbol{w}w^{(hi)}$ (84.7%) is due to the blank pixels being encoded at a 10-Hz spike rate rather than being left inactive.

Table 4 presents a comparison of the performance of eWB-eRBP with that of relevant works using limited-precision weights ($\leq$ 8-bit). For a fair comparison, we chose event-based algorithms applied to fully connected SNNs. Notably, most of them use higher precision than 1-bit.

TABLE 4. Comparison of reported classification accuracy of quantized fully connected SNNs on the MNIST dataset.

Learning algorithm	Structure	Weight precision	Accuracy (%)
CD + BP [23]	784-500-500-10	4-bit	91.35
STDP + eCD [16]	784-500-10	4-bit	94.80
eRBP [9]	784-200-200-10	8-bit	96.50
STDP + BP [7]	784-6400-10	1-bit + 24-bit	95.70
STDP + BP [8]	784-6400-10	1-bit + 32-bit	92.14
eWB-eRBP (This work)	784-1000-1000-10	1-bit	95.35

Nevertheless, the classification accuracy is lower than or only slightly better than that of our work. This highlights the performance of eWB.

The works by Yousefzadeh *et al.* [7] and Srinivasan and Roy [8] partly use 1-bit weights, but the usage is limited to only the weights between the input and hidden layers. The weights between the hidden and output layers are of higher precision to minimize the classification accuracy loss. Therefore, eWB is the first event-driven weight binarization algorithm with locality that ensures high performance.

The accuracy loss caused by the use of binary weights is compensated for to some extent by increasing the number of hidden neurons as shown in the comparison with eRBP in Tables 2 and 4. The same holds for DNNs with binary weights, e.g., BNN [18] and BinaryConnect [20], which include much more parameters than full-precision weight networks to achieve comparable performance. From a perspective of memory usage, one should consider the total memory usage for trainable parameters in the networks with binary weights in comparison with full-precision weight networks to estimate the gain in memory-efficiency.

We can estimate the additional computation for eWB to eRBP from Eq. (12). The weight update evaluation Δw_{ij} requires one multiplication ($\lambda_{ij} \times w_{ij}$) and one subtraction ($T_i - 2\lambda_{ij} w_{ij}$) operations in addition to eRBP. The evaluation of the Lagrange multiplier update requires one subtraction ($1 - w_{ij}^2$) and three multiplication ($w_{ij} \times w_{ij}$, $\left(1 - w_{ij}^2\right) \times \Theta$, $\eta_\lambda \times \left(1 - w_{ij}^2\right) \Theta$). Thus, the computational cost of a single update is two substraction and four multiplication operations in addition to eRBP.

The eWB-eRBP algorithm is an example to demonstrate the compatibility of eWB with event-based learning algorithms. In principle, eWB can also be combined with other event-based learning algorithms with appropriate modifications and can serve as a common weight binarization algorithm for various event-based learning algorithms. In this regard, attention should be paid to the performance reduction resulting from optimal weight binarization instead of the absolute performance when evaluating the performance of eWB. This is because the absolute performance

is mainly determined by the learning algorithm combined with eWB.

Although we have applied LMM to weight binarization in this study, any other constraints can be considered as long as they are mathematically well-defined. For instance, ternary weight (0, ±1; 2-bit precision) constraints with eRBP can be formulated as $g\left(w_{ij}\right) = w_{ij}(1-w_{ij}^2)\Theta(I_i)s_j^{pre}(t)$ instead of Eq. (10). This constraint function outputs zero when $w_{ij} = 0$ or $w_{ij} = \pm 1$, enabling the algorithm to learn optimal ternary weights. Further, 3-bit weight (0, ±1, ±2, ±3) constraints can be considered using the constraint function $g\left(w_{ij}\right) = w_{ij}(1-w_{i??}^2)(4-w_{ij}^2)(9-w_{ij}^2)\Theta(I_i)s_j^{pre}(t)$. Therefore, the proposed LMM-based learning algorithm forms the foundation for event-based learning with various constraints.

The use of limited-precision weights improves not only memory efficiency but also energy efficiency. Energy efficiency is a key attribute of neuromorphic computing and is a defining motivation for event-based learning algorithms as alternatives to layer-wise synchronous learning such as BP. In digital neuromorphic hardware, a lower precision of the data format reduces the energy consumed in arithmetic operations. Horowitz [29] identified a 30-fold (18.5-fold) improvement in the energy efficiency by replacing 32-bit floating-point data with 8-bit fixed-point data in addition (multiplication) operations. The use of binary weights completely avoids the multiplication of weights and low-pass filtered spikes, which are otherwise needed for every synaptic operation. Given that synaptic operations impose the most significant workload on neuromorphic hardware, as is the case for multiply-accumulate operations in DNNs [16], SNNs with binary weights can achieve a large improvement in energy efficiency. Nevertheless, the degree of improvement depends on neuromorphic hardware design, which is not specified in this study.

VI. CONCLUSION

In this study, we proposed an eWB algorithm that optimally binarizes weights in an SNN based on local events. The optimal configuration of binary weights is calculated using the LMM with binary weight constraints. Given that eWB addresses local data only to update weights in an event-based manner, it is inherently compatible with multicore neuromorphic hardware. When combined with an event-based learning algorithm using an appropriate loss function, eWB enables the network to learn binary weights that minimize the loss function. This was demonstrated using eWB-eRBP (eWB combined with eRBP), which was applied to train fully connected SNNs on MNIST. The consequent classification accuracy is 95.35%, whereas eRBP with 32-bit weights yielded an accuracy of 97.20%. The results indicate an accuracy reduction of 1.85% as the cost of optimal weight binarization. To the best of our knowledge, eWB is the first method to learn binary weights based on events; therefore, a comparison with directly related methods is unavailable at the moment. Nevertheless, to highlight the importance of optimal binary weights in performance, eWB-eRBP was compared with fWB-eRBP (with non-optimal binary weights that were forcibly binarized) and was shown to yield better performance and more stable performance evolution over the training epoch than fWB-eRBP.

Finally, eWB is scalable to any event-based learning algorithm with appropriate modifications, thus serving as a common weight binarization method. The LMM is also scalable to any weight constraint as long as the constraint functions are mathematically well-defined. The eWB algorithm is an example that demonstrates this scalability.

APPENDIX

The eRBP algorithm is a presynaptic event-driven local learning rule that uses direct feedback alignment (Fig. 1). In eRBP, the weight update with a mean-squared loss function is formulated as

$$\Delta w_{ij}(t) = -T_i(t)\Theta(I_i)s_j^{pre}(t), \tag{13}$$

which realizes a three-factor rule with (i) presynaptic spike (s_j^{pre}), (ii) postsynaptic signal Θ, corresponding to the derivative of the postsynaptic activation, and (iii) error signal T_i, which backpropagates through random feedback channels.

(i) The presynaptic spikes are the output of neuron i, which is modeled using an LIF model that includes two defining variables, namely, the synaptic current I_i and subthreshold somatic membrane potential V_i:

$$\begin{cases} \tau_{syn}\dfrac{d}{dt}I_i = -I_i + \sum_j w_{ij}s_j(t)\xi(t) \\ \mathrm{C}\dfrac{d}{dt}V_i = -g_V V_i + I_i, \end{cases}$$

where w_{ij}, s_j, and ξ denote the weight between neurons j and i, spikes from neuron j, and a stochastic Bernoulli process with probability $(1-p)$, respectively. The time constant for the synaptic current is denoted by τ_{syn}. The ion conductance through the membrane is denoted as g_V.

(ii) As a workaround for the postsynaptic activation being non-differentiable, the derivative of the postsynaptic activation is approximated as a boxcar function Θ:

$$\Theta(I_i) = \begin{cases} 1 & \text{if } b_{min} < I_i < b_{max} \\ 0 & \text{otherwise.} \end{cases}$$

This corresponds to the derivative of a hard sigmoid function with two transition points ($b_{\min}$ and $b_{\max}$).

(iii) The error signal T_i is formulated as

$$T_i(t) = \sum_k e_k(t)\, g_{ik}, \tag{14}$$

where e_k is the error signal from the error-coding neuron k. The constant g_{ik} denotes the fixed random feedback weight from the error-coding neuron k to the hidden neuron i. It is noteworthy that this error signal is non-local to the synapse w_{ij}, and thus unavailable for updating the weight w_{ij} using Eq. (13). It is conceivable that the data may be moved from the location of error evaluation to the synapse during updating; however, this is not an optimal strategy

for neuromorphic hardware in which neurons communicate using events only. To render the error local to the target synapses, eRBP uses two error-coding neurons with somatic potentials V^{E+} and V^{E-} for each output dimension. They code for false positive and negative errors, respectively. Their subthreshold behaviors are modeled using a perfect integrate-and-fire model

$$C\frac{d}{dt}V^{E\pm} = \pm w^E(s^P(t) - s^L(t)),$$

where s^P and s^L are the spike trains from the prediction neurons and labels, and w^E is a positive constant. The false positive error coding neuron (potential V^{E+} and weight w^E) spikes and generates the spike train s^{E+} when $s^p = 1$ and $s^L = 0$, whereas the false negative error coding neuron (potential V^{E-} and weight $-w^E$) spikes and generates the spike train s^{E-} when $s^p = 0$ and $s^L = 1$. The consequent spike trains s_j^{E+} and s_j^{E-} (from the two error-coding neurons for label j), rather than the error data themselves, are relayed to the target synapses through the random weight g_{ij} so that the communication architecture is well suited for neuromorphic hardware.

The error spike trains s_j^{E+} and s_j^{E-} from label j are subsequently encoded as firing rates to eventually realize the error signal T_i in Eq. (14). To this end, each neuron in the output and hidden layers is given a dendritic compartment that calculates the dendritic potential (U_i^h for hidden neuron i and U_i^p for prediction neuron i) using a leaky integrated model

$$C\frac{d}{dt}U_i^h = -g_U U_i^h + \sum_j g_{ij}(s_j^{E+}(t) - s_j^{E-}(t)),$$

and

$$C\frac{d}{dt}U_i^p = -g_U U_i^p + w^E(s_i^{E+}(t) - s_i^{E-}(t)).$$

This dendritic potential is equivalent to the error signal T_i and is local to each target synapse. Therefore, the learning rule in Eq. (13) can be rewritten as

$$\Delta w_{ij} = \eta U_i \Theta(I_i) s_j(t).$$

The parameters used in this study are listed in Table 1.

REFERENCES

[1] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers Neurosci.*, vol. 12, p. 774, Oct. 2018.

[2] D. S. Jeong, "Tutorial: Neuromorphic spiking neural networks for temporal learning," *J. Appl. Phys.*, vol. 124, no. 15, Oct. 2018, Art. no. 152002.

[3] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, Nov. 2019.

[4] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers Neurosci.*, vol. 13, p. 95, Mar. 2019.

[5] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation in state-of-the-art deep neural network architectures," 2019, *arXiv:1903.06379*. [Online]. Available: http://arxiv.org/abs/1903.06379

[6] G.-Q. Bi and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, Dec. 1998.

[7] A. Yousefzadeh, E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "On practical issues for stochastic STDP hardware with 1-bit synaptic weights," *Frontiers Neurosci.*, vol. 12, p. 665, Oct. 2018.

[8] G. Srinivasan and K. Roy, "ReStoCNet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing," *Frontiers Neurosci.*, vol. 13, p. 189, Mar. 2019.

[9] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers Neurosci.*, vol. 11, p. 324, Jun. 2017.

[10] D. Kim, V. Kornijcuk, C. S. Hwang, and D. S. Jeong, "SPSNN: Nth order sequence-predicting spiking neural network," *IEEE Access*, vol. 8, pp. 110523–110534, 2020.

[11] F. Ponulak and A. Kasinski, "Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, Feb. 2010.

[12] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing–based decisions," *Nature Neurosci.*, vol. 9, no. 3, pp. 420–428, Mar. 2006.

[13] S. M. Bohte, J. N. Kok, and H. L. Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, Oct. 2002.

[14] M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and C. DeSalvo, "Bio-inspired stochastic computing using binary CBRAM synapses," *IEEE Trans. Electron Devices*, vol. 60, no. 7, pp. 2402–2409, Jul. 2013.

[15] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers Neurosci.*, vol. 7, p. 272, Jan. 2014.

[16] E. O. Neftci, B. U. Pedroni, G. Cauwenberghs M. Al-Shedivat, and S. Joshi, "Stochastic synapses enable efficient brain-inspired learning machines," *Frontiers Neurosci.*, vol. 10, p. 241, Jun. 2016.

[17] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*. [Online]. Available: https://arxiv.org/abs/1602.02830

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2016, pp. 525–542.

[20] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[21] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[22] S. Lu and A. Sengupta, "Exploring the connection between binary and spiking neural networks," 2020, *arXiv:2002.10064*. [Online]. Available: http://arxiv.org/abs/2002.10064

[23] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S.-C. Liu, "Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms," *Frontiers Neurosci.*, vol. 9, p. 222, Jul. 2015.

[24] M. Maher and J.-F. Puget, *Principles and Practice of Constraint Programming-CP98: 4th International Conference, CP98, Pisa, Italy, Oct. 26-30, 1998, Proceedings*. Berlin, Germany: Springer, 1998.

[25] J. C. Platt and A. H. Barr, "Constrained differential optimization," presented at the Int. Conf. Neural Inf. Process. Syst., 1987.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[27] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[28] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," in *Proc. 7th Annu. Neuro-Inspired Comput. Elements Workshop (NICE)*, 2019, pp. 1–8.

[29] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 10–14.

DOHUN KIM received the B.S. degree in materials science and engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree. Since 2016, he has been focused on learning algorithms for neuromorphic hardware implementation, especially for temporal learning.

GUHYUN KIM received the B.S. and Ph.D. degrees in material science and engineering from Seoul National University, Seoul, South Korea, in 2015 and 2020, respectively. He is currently a Postdoctoral Scholar with Hanyang University, Seoul. His research interests include learning algorithms for spiking neural networks and deep neural networks.

CHEOL SEONG HWANG received the Ph.D. degree from Seoul National University, Seoul, South Korea, in 1993. Since 1998, he has been a Professor with the Department of Materials Science and Engineering, Seoul National University. His current research interests include high-k gate oxides, dynamic random access memory capacitors, new memory devices, including resistive RAM devices and ferroelectric materials and devices, energy storage capacitors, and neuromorphic computing.

DOO SEOK JEONG (Member, IEEE) received the B.E. and M.E. degrees in materials science from Seoul National University, in 2002 and 2005, respectively, and the Ph.D. degree in materials science from RWTH Aachen University, Germany, in 2008. He was with the Korea Institute of Science and Technology, from 2008 to 2018. He is currently an Associate Professor with Hanyang University, South Korea. His current research interests include spiking neural networks for sequence learning and future prediction, learning algorithms, spiking neural network design, and digital neuromorphic processor design.